Development and Validation of a Hierarchical Memory Model Incorporating CPU- and Memory-Operation Overlap

Yong Luo, Olaf M. Lubeck, Harvey Wasserman
Federico Bassetti, Kirk W. Cameron
Los Alamos National Laboratory
Mail Stop B256
Los Alamos, NM 87545
{yongl, oml, hjm, fede, kirk}@lanl.gov

Abstract

In this paper, we characterize application performance with a "memory-centric" view. Using a simple strategy and performance data measured on actual machines, we model the performance of a simple memory hierarchy and infer the contribution of each level in the memory system to an application's overall cycles per instruction (cpi). Included are results affirming the usefulness of the memory model over several platforms, namely the SGI Origin 2000, SGI PowerChallenge, and the Intel ASCI Red TFLOPS supercomputers. We account for the overlap of processor execution with memory accesses - a key parameter, which is not directly measurable on most systems. Given the system similarities between the Origin 2000 and the PowerChallenge, we infer the separate contributions of three major architecture features in the memory subsystem of the Origin 2000: cache size, outstanding loads-under-miss, and memory latency.

Keywords: *performance evaluation, cache, memory subsystem, computer architecture, and microprocessor*

1. Introduction

The performance and scalability of high performance scientific applications on large-scale parallel machines are more dependent on the hierarchical memory subsystems of these machines than the peak instruction rate of the processors employed [1-2]. A few attempts [14, 15] have been made to characterize the memory system performance impact on the total runtime. Many architectural improvements such as out-of-order execution and more outstanding misses are widely studied by simulations. However, studying the performance impact of memory subsystem and processor architecture improvements based on real applications on production machines is rarely attempted. Such studies promise insight for engineers in future architectural design improvements and will provide information that software engineers can use to improve code performance in scalable environments.

In this paper, we model real application performance with a "memory-centric" view. The applications and their realistic problem sizes are a representative part of the Los Alamos National Laboratory (LANL) computational physics workload and most have been designed with referential locality in mind. Instruction-level simulation of even small problem sets would require at least 12-36 hours and we thus resort to experimental techniques and modeling to understand the effect of changes in major architectural parameters. Using overall average effect strategy and empirical performance data from hardware performance counters, we infer the contribution of each level in the memory system to the application's overall cycles per instruction (cpi). We account for the overlap of processor execution with memory accesses - a key performance parameter that is not directly measurable on most systems.

Performance data on the application codes are obtained on the latest Origin 2000 systems and Power Challenge machines from SGI, along with the Intel ASCI Red TFLOPS machine. This paper discusses only single node executions. The SGI machines provide a unique performance evaluation opportunity since the architectures employ identical R10K processors but differ significantly in the design of the memory subsystems so that performance studies due solely to the memory architecture are possible. The same executables are used on both Origin2000 and PowerChallenge to

WOSP98, Santa Fe, N.M. 1998 ACM 1-58113-060-0/98/00.

eliminate software difference. In particular, there are three major memory architecture differences: 1) secondary cache size, 2) latencies to the main memory, and 3) number of outstanding cache misses. Thus, we are able to infer the separate contribution of each of these on the performance of the application benchmarks.

The following sections of this paper describe: the parts of the machine architecture relevant to this work, small descriptions of the codes from the Los Alamos National Lab computational physics workload, the model and empirical methodology, validation of the model, results, analysis and major conclusions.

2. Architecture descriptions

2.1. Origin 2000 and PowerChallenge

The PowerChallenge is an SMP architecture that employs a central bus to interconnect memories and processors [3]. The bus bandwidth (1.2 Gbytes/sec) does not scale with more processors. Cache coherence is maintained through a snoopy bus protocol which broadcasts cache information to all processors connected to the bus. The Origin 2000, on the other hand, is a distributed shared memory (DSM) architecture which uses a switch interconnect that improves scalability by providing interconnect bandwidth proportional to the number of processors and memory modules [4]. Coherence is maintained by a distributed directory-based scheme. Figure 1 shows a network view of the machine. Each router in the hypercube topology connects two nodes to the network. Each node contains two processing elements and one local memory unit. A 128-processor system, for example, consists of a fifth-degree hypercube with 4 processors per router.

The processing elements of both the Origin 2000 and PowerChallenge systems use a 200MHz MIPS R10000 microprocessor. The processor is a 4-way super-scalar architecture which implements a number of innovations to reduce pipeline stalls due to data starvation and control flow [5]. For example, instructions are initially decoded in-order, but are executed out-of-order. Also, speculative instruction fetch is employed after branches. Register renaming minimizes data dependencies between floating-point and fixed-point unit instructions. Logical destination register numbers are mapped to the 64 integer and 64 floating point physical registers during execution. The two programmable performance counters track a number of events [6] and were a necessity for this study. The most common instructions typically have one- or two-clock latencies.

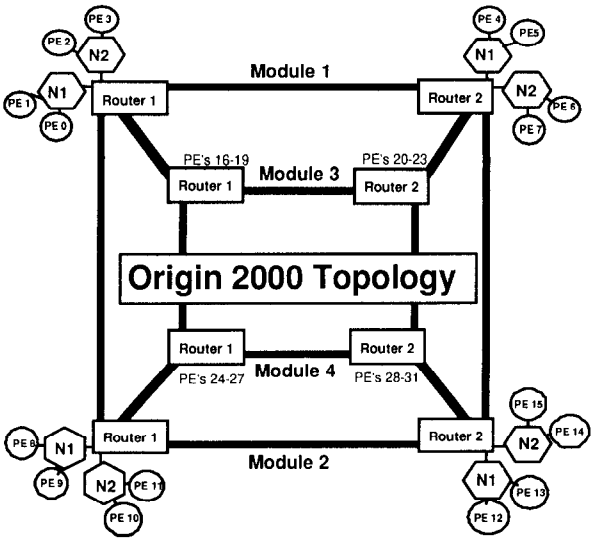

Figure 1. Origin 2000 topology for 32-proc system

While the processing elements of the PowerChallenge and Origin 2000 systems are identical, there are major differences in the memory architecture and corresponding performance of the two systems. The PowerChallenge is an UMA architecture with a latency of 205 clocks (1025 ns). Latencies to the memory modules of the Origin 2000 system, on the other hand, depend on the network distance from the issuing processor to the destination memory node. Accesses issued to local memory take about 80 clocks (400 ns) while latencies to remote nodes are the local memory time plus 33 clocks for an off-node reference plus 22 clock periods (CP; 110 ns) for each network router traversed. In the case of a 32 processor machine, the maximum distance is 4 routers, so that the longest memory access is about 201 clocks (1005 ns) which is close to the uniform latency of the PowerChallenge. This unique feature of Origin 2000 systems provides us a good opportunity to adjust the memory access latency by placing memory and execution thread on different nodes.

In addition, improvements in the number of outstanding loads that can be queued by the memory system were made. Even though the R10000 processor is able to sustain four outstanding primary cache misses, external queues in the memory system of the PowerChallenge limited the actual number to less than two. In the Origin 2000, the full capability of four outstanding misses is possible. The L2 cache sizes of these two systems are also different. A processor of PowerChallenge can be equipped up to 2MB L2 cache while a CPU of Origin 2000 system always has a L2 cache of 4MB.

2.2. Intel ASCI Red TFLOPS Supercomputer

The ASCI Red Supercomputer is a Massively Parallel Processor (MPP) with a distributed memory Multiple-Instruction, Multiple Data (MIMD) architecture [16]. All aspects of this system architecture are scalable, including communication bandwidth, main memory, internal disk storage capacity, and I/O [17]. The ASCI Red maintains communication through an Interconnection Facility (ICF) in a 38x32x2 topology with a peak (sustainable) bi-directional bandwidth of 800 MB/sec [16]. A Kestrel board holds two compute nodes connected through a Network Interface Chip (NIC) and attached to a Mesh Router Chip (MRC). The memory subsystem on an individual compute node is implemented using the Intel 82453 Chipset with 128 MB/node.

ASCI Red is composed of 9,216 processors providing 4,536 compute nodes. Each compute node consists of two 200 MHz Pentium Pro Processors. The 200 MHz Pentium Pro processor is a 3-way super-scalar architecture that reduces pipeline stalls utilizing features such as out-of-order execution, speculative execution of branches, and register renaming. Two programmable performance counters are also available, providing the data used in this study. Each processor includes separate 8KB data and instruction caches along with 256KB secondary L2 cache. This L2 cache is located on a separate die in the same package closely coupled via a dedicated 64-bit full-clock-speed backside cache bus. The L1 data cache can handle as many as four outstanding misses and has a miss latency of three cycles, whereas the L2 cache miss latency is about 50 cycles [15]. Only one CPU on a node is used for this experiment.

3. LANL benchmark code information

Four applications that form the building blocks for many nuclear physics simulations were used in this study. A performance comparison of the Origin and PowerChallenge architectures has been done using these codes [7].

3.1. Code descriptions

SWEEP3D is a three dimensional solver for the time independent, neutral particle transport equation on an orthogonal mesh [8]. In SWEEP3D, the main part of the computation consists of a "balance" loop in which particle flux out of a cell in three Cartesian directions is updated based on the fluxes into that cell and on other quantities such as local sources, cross section data, and geometric factors. The cell-to-cell flux dependence, i.e., a given cell cannot be computed until all of its upstream neighbors have been computed, implies a recursive or wavefront structure. The specific version used in these tests was a scalar-optimized "line-sweep" version [Koch] that involves separately nested, quadrant, angle, and spatial-dimension loops. In contrast with vectorized plane-sweep versions of SWEEP3D, there are no gather/scatter operations and memory traffic is significantly reduced through "scalarization" of some array quantities. Because of these features, L1 cache reuse on SWEEP3D is fairly high (the hit rate is about 85%). A problem size of N implies N^3 grid points.

HYDRO is a two-dimensional explicit Lagrangian hydrodynamics code based on an algorithm by W. D. Schulz [9]. HYDRO is representative of a large class of codes in use at the Laboratory. The code is 100% vectorizable. An important characteristic of the code is that most arrays are accessed with a stride equal to the length of one dimension of the grid. HYDRO-T is a version of HYDRO in which most of the arrays have been transposed so that access is now largely unit-stride. A problem size of N implies N^2 grid points.

HEAT solves the implicit diffusion PDE using a conjugate gradient solver for a single timestep. The code was written originally for the CRAY T3D using SHMEM. The key aspect of HEAT is that its grid structure and data access methods are designed to support one type of adaptive mesh refinement (AMR) mechanism, although the benchmark code as supplied does not currently handle anything other than a single-level AMR grid (i.e. the coarse, regular level-1 grid only). A problem size of N implies N^3 grid points.

NEUT is a Monte-Carlo particle transport code. It solves the same problem as SWEEP3D but uses a statistical solution of the transport equation. Particles are individually tracked through a three dimensional mesh where they have some probability of colliding with cell material. The output from the particle tracking is a spatial flux discretized over the mesh. Vector (or data parallel) versions of this type of code exist which track particle ensembles rather than individual ones. A problem size of N implies N^3 grid points and 10 particles per grid point.

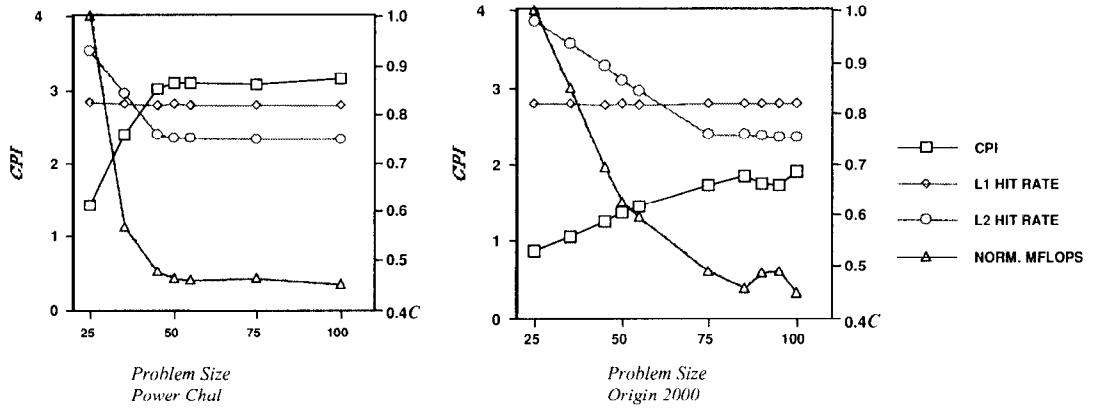

Figure 2. Performance of HEAT as a function of linear problem size. The right axis shows cache hit rates and normalized MFLOPS.

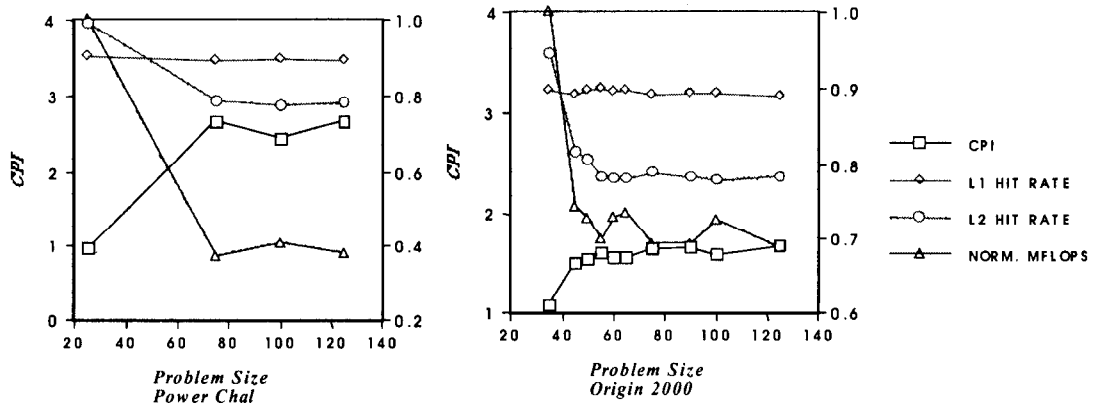

Figure 3. Performance of SWEEP as a function of linear problem size

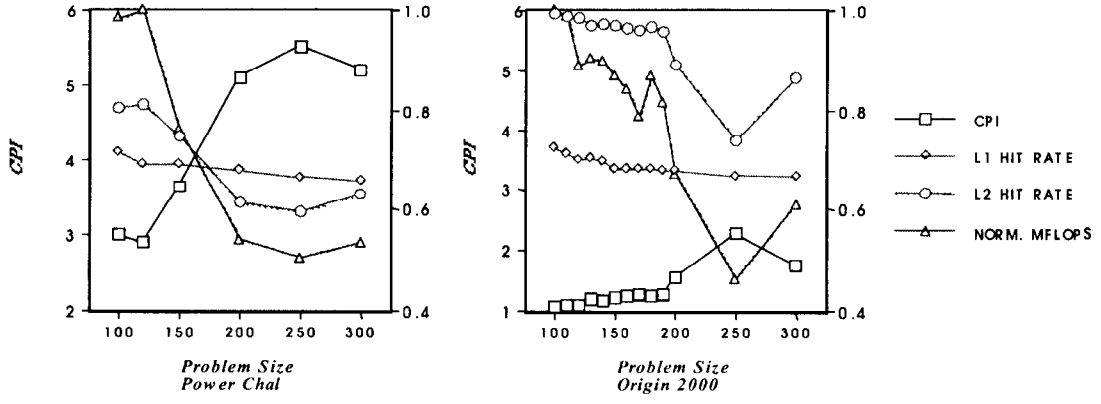

Figure 4. Performance of HYDRO as a function of linear problem size.

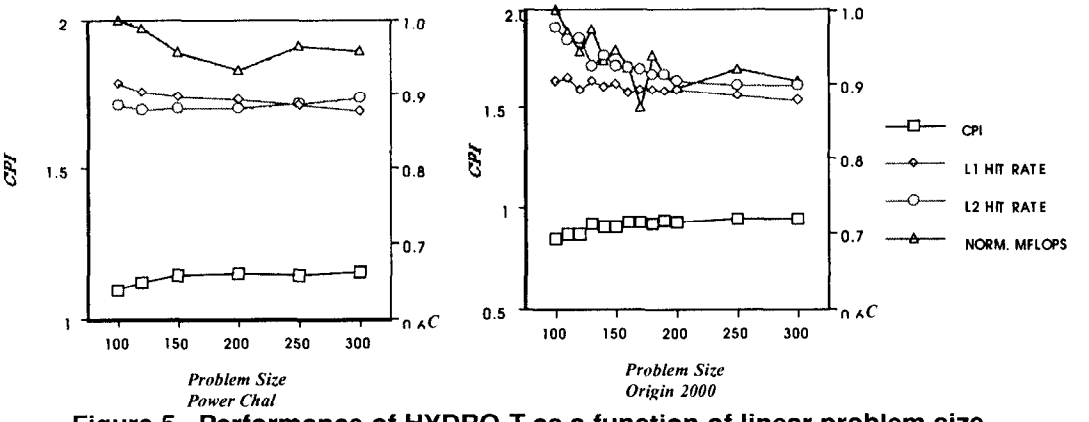

Figure 5. Performance of HYDRO-T as a function of linear problem size.

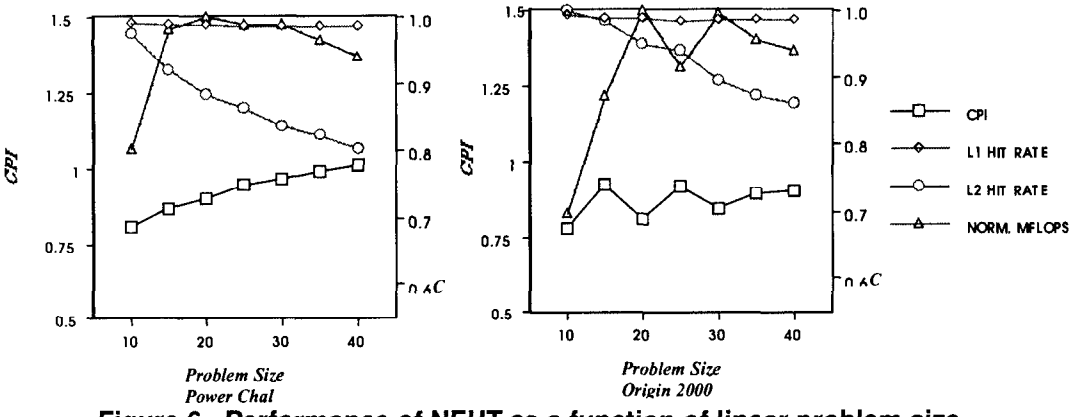

Figure 6. Performance of NEUT as a function of linear problem size.

3.2. Performance characteristics

In this section we present some single-processor characteristics of the benchmark codes as obtained from performance counters on the Origin 2000. Note that the maximum MFLOPS observed may, in some cases, be obtained from unreasonably-small problem sizes relative to actual production runs; the data are presented here merely as a reference for the normalized Mflop curves in Figures 2-6.

Detailed performance characteristic data for these codes were collected on a 1-MB L2 PowerChallenge system and a 4-MB L2 Origin2000 system. Performance data as a function of problem size for the Power Challenge and Origin are illustrated in Figures 2 through 6. MFLOPS curves are normalized such that the maximum rate for each code is one.

The codes' overall *cpi* curves are generally the inverse of their corresponding MFLOPS curves; that is, an increasing *cpi* corresponds to a decreasing MFLOPS at nearly the same slope and vice versa. The *cpi* of three of the codes (HEAT, HYDRO and SWEEP) is strongly dependent on problem size.

The above figures show that normalized MFLOPS curves (except for HYDRO-T) follow the tendencies of the L2_hit curves. On the PowerChallenge system, a drop in L2_hit rate causes much more impact to MFLOPS than it does on the Origin system. This is due to lower memory latency (both actual and effective) on the Origin2000 system. Although not shown in the figures, we calculated TLB hit ratio and branch prediction hit ratio. The calculation shows that MIPS R10000 processor can do a good job of speculative branch prediction. All four benchmark codes (HEAT, HYDRO, HYDRO-T and SWEEP) have branch prediction hit ratios over 99%. This means that over 99% of speculated branch predictions are taken in real executions. TLB hit ratios for all these codes are higher than 98%. This high TLB hit ratio implies that the impact of TLB misses can be ignored for these data sets.

4. Model description

The analysis in the following sections uses a simplified mean value parameterization [11] to separate CPU execution time from stall time due to memory loads/stores. Figure 7 is a pictorial description of the times in the model.

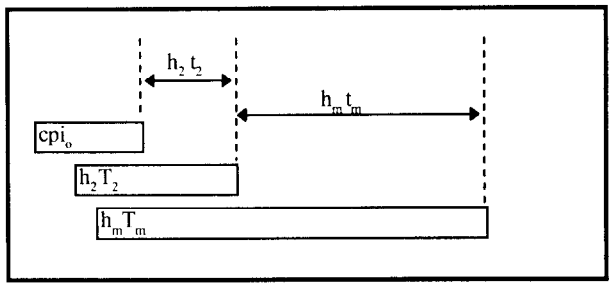

Figure 7. Relationship of modeled times

The model projects the overall *cpi* of an application as a function of CPU execution time and average memory access times:

$$cpi = cpi_0 + \sum_{i=2}^{nlevels} h_i * t_i \qquad (1)$$

where cpi_0 is defined to be the *cpi* of the application assuming that all memory accesses are from an infinite L1 cache and take 1 CP (i.e., the i=1 term is included in cpi_0), and h_i and t_i are, correspondingly, the hits per instruction and average non-overlapped access times for the ith level in the memory hierarchy. Measured access times at the ith level correspond to access time from level i to the registers. The second term of Eq. 1 is also referred to as cpi_{stall}.

If no overlap of CPU execution and memory accesses occur, every memory access to the ith level incurs the full round-trip latency, which we denote as T_i. We define (following Larson [12]) a measure of the overlap of memory accesses with computation as m_0, where

$$cpi = cpi_0 + (1 - m_0) \sum_{i=2}^{nlevels} h_i * T_i \qquad (2)$$

and, from Eq 1, m_0 is one minus the ratio of the average memory access time to the maximum memory access time:

$$m_0 = 1 - \frac{\sum_{i=2}^{nlevels} h_i * t_i}{\sum_{i=2}^{nlevels} h_i * T_i} \qquad (3)$$

We note here that the separation of computational time from memory access time in this model implies that the two can be treated independently (i.e., that cpi_0 is constant). In fact, the out-of-order execution of the R10000 processor means that different dynamic instruction sequences will be seen for different size problems. The assumption that this effect is small is tested with an R10000 simulator in a later section.

The effect of increasing the round-trip memory latency to $T_m + dT_m$ is depicted in Fig. 8. Once the latency hiding ability of the architecture on a particular code has been exhausted, any additional main memory latency will simply add to the non-overlapped time t_m. In this case, the new *cpi* (from Eq. 1, where the sum is over the L2 cache and main memory) will be:

$$cpi' = cpi_0 + h_2 t_2 + h_m (t_m + dT_m) \qquad (4)$$

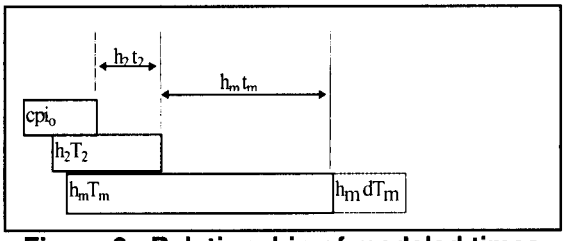

Figure 8. Relationship of modeled times

This equation predicts a linear relationship between dT_m and cpi' of slope h_m. If any additional memory latency incurred by dT_m can be hidden, the increase in *cpi* will be strictly less than that predicted by Equation 4.. That is, the relationship is an upperbound for the increase in time due to memory latency. This analysis will be used and verified in a later section.

5. Measurements and validation

5.1. Measurements

The model described in the previous section provides the foundation for an analysis of the Origin 2000's architectural features on application performance.

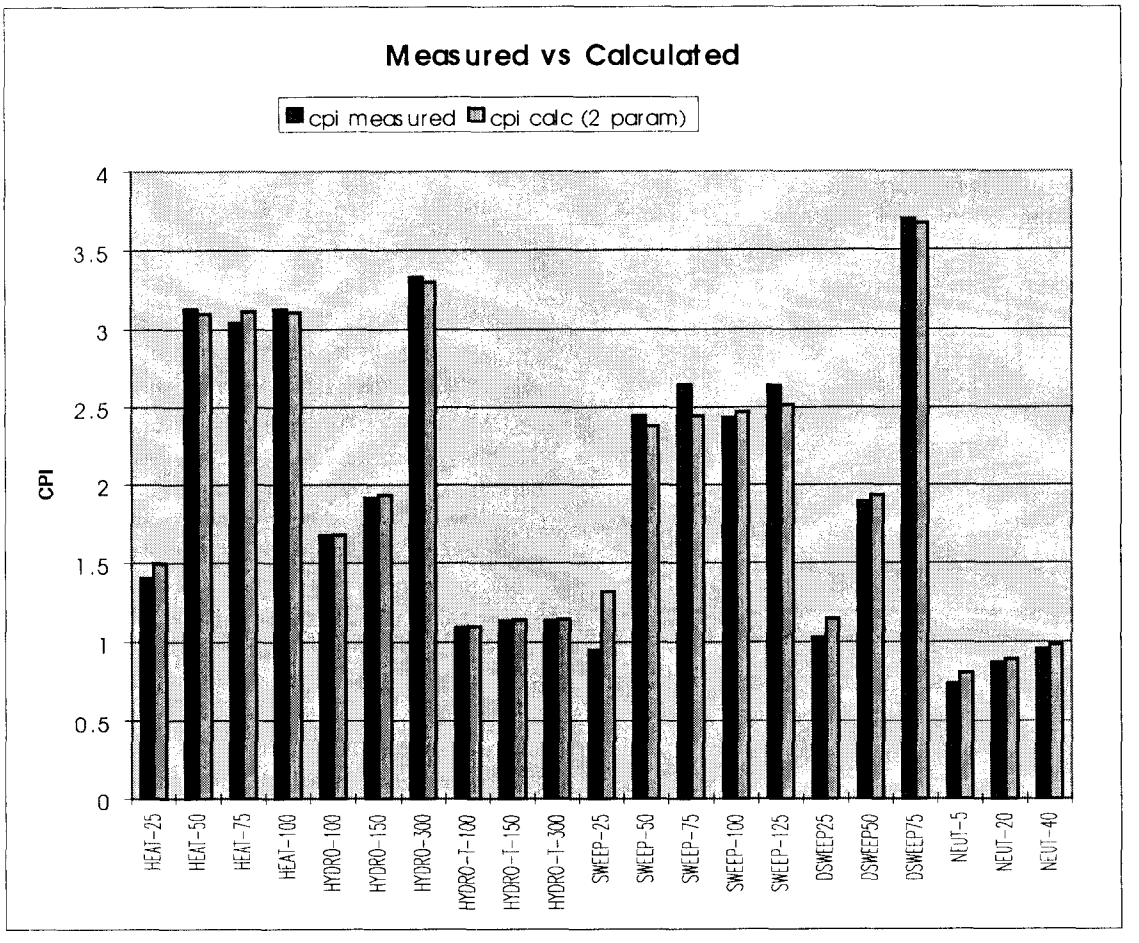

Figure 9. Model fit for codes with varying problem sizes

The first key issue is determination of the amount of memory access time that is overlapped by computation. Although this overlap is not directly measurable using the R10000 performance counters, we can infer the overlap for an individual application by fitting empirical performance data obtained from its execution using different problem sizes.

R10000 performance counters supply measurements of the total execution cycles and total graduated instructions. The ratio of these two measurements gives the overall cpi of the application. The hit ratios (coming from the same application executing on different problem sizes) are also directly measurable and the unknowns in Equation 1 become the average times, t_i, and cpi_0. The value of cpi_0 can be obtained by measuring the *cpi* of a problem that fits entirely in the L1 cache. The remaining unknowns are inferred from the measured data by a least squares fit constrained such that

$$0 <= t_i <= T_i,$$

Table 1 shows the model parameters for each of the LANL benchmark codes determined from a data set of executions on the 1-MB L2 PowerChallenge. The least square fit generally has errors that are less than 6%. The maximum latencies, T_i, are measured with LMBENCH [10] and are found to be consistent with numbers published by SGI.

Table 1. Model parameters for each code (PowerChallenge)

	t_2	t_m	cpi_0
HEAT	2	128	0.74
HYDRO	3	117	0.89
HYDRO-T	0	69	.9
SWEEP	11	145	.88
NEUT	2.2	205	.77

5.2. Validation

Validation of the inferred model parameters is accomplished using the model to predict performance on

Table 2. Memory access times, t_i, for the PowerChallenge, Origin 2000, and Intel ASCI Red

	t_2 Power Chal	t_m Power Chal	t_2 Origin 2000	t_m Origin 2000	t_2 Intel Red	t_m Intel Red
HEAT	2	128	0	60		
HYDRO	3	117	2.4	50	2	20
HYDRO-T	0	69	0	11	5.1	5.8
SWEEP	11	145	11	43		
NEUT	2.2	205	11	80		
LMBENCH	11	205	11	80	7	37

a different machine configuration. Original data from a PowerChallenge with a 1-MB secondary cache is used to determine the unknown model parameters which are then used to predict the performance of each code on a 2-MB PowerChallenge. Figure 9 shows that the fit is extremely close.

6. Results and analysis

6.1. Analysis of stall time due to memory accesses

Table 2 compares the memory access times, t_i, for the benchmark codes on the PowerChallenge, the Origin 2000, and the Intel ASCI Red supercomputer. In general, L2 cache accesses are completely overlapped with computation (low values of t_2) for the comparable Origin 2000 and PowerChallenge. Additionally, the observed values of t_m suggest that about one-half of the main memory latency is hidden on the PowerChallenge, and Origin 2000. The exception is SWEEP (not measured on Intel Red) where the value of 11cps for t_m indicates that accesses to the secondary cache are not overlapped. The reason that SWEEP stands out may be due to loop-carried dependencies in the inner loops. These dependencies present less prefetch opportunities for the compiler and result in less overlap of processor execution with memory accesses. We believe that the model parameters for NEUT may be inaccurate. There is so little time associated with the memory accesses for NEUT (due to high cache-hit ratios; see Figure 6) that small absolute least square errors can result in large relative changes to the parameters.

Table 2 also shows effects attributed to the number of usable registers on the two different microprocessors, namely the MIPS R10000, and Intel Pentium Pro. The 200MHz R10000 provides 64 registers [5a] whereas the 200MHz Pentium Pro allows at most 40 registers for general use [15]. This gap in registers available degrades overlap performance as expected leading to a higher percentage of overlap work performed by both the Origin 2000 and PowerChallenge. This is directly confirmed by the higher percentage of non-overlapped access time (out of nominal full latency) for HYDRO on Intel ASCI Red in both L2 and memory levels.

Figures 10, 11, and 12 show graphs of cpi_{stall} relative to the overall cpi for all machines on most codes. The second half of each figure shows the corresponding overlap parameter, m_0. A number of general observations are apparent from the graphs. The overall cpi on the Origin is typically less by factors of up to three (see also Luo, et al. [7]) on the PowerChallenge and consistently less than those measured on the Intel ASCI Red. The percentage of cpi represented by stall time on the Origin can be less than 40%, while, on the PowerChallenge, it can be as large as 80%. Two codes, HYDRO-T and NEUT, exhibit high locality of reference and cpu stalls due to memory accesses are less than 10% of the total time. A study of the algorithms/implementations of these codes would lead one to expect this. NEUT has a modest number of scalar variables per particle that are used many times before another particle is computed (high temporal locality). HYDRO-T is a 2D code and was re-coded from the original HYDRO so inner loops have stride-1 vectorizable loops (high spatial locality). The success of the transposition can be seen by comparing each version in the figures.

Memory overlap parameters are higher on the Origin than on the PowerChallenge, indicative of the better latency hiding capability of the Origin. As discussed previously, and confirmed by the overlap parameters, the Intel ASCI Red maintains an even lower hiding capability than both the Origin and the PowerChallenge. Two extreme examples are given: HYDRO-T with very high overlap, and SWEEP (not shown for the Intel ASCI Red), with very low overlap. The high spatial locality of HYDRO-T means that there is a great deal of parallelism between L1, L2 and main memory accesses. Additionally, on the Origin 2000, major portions of this 2-D algorithm fit entirely in the 4-MB L2 cache. In contrast, SWEEP shows much less overlap on either the PowerChallenge or the Origin. This is consistent with the information in Table 2 in which we attributed to loop-carried dependencies. The results for NEUT, where

the PowerChallenge shows high overlap and the Origin shows very low overlap, are again due to the large parameter changes associated with the least-squares fit mentioned above.

6.2. Separate contributions to the stall time.

As described in Section 1, we performed an experiment in which we systematically varied T_m, the latency to main memory seen by an executing thread, by placing the thread and its associated data on two different nodes of the Origin. Figures 13 displays the measurements for HEAT (HYDRO, HYDRO-T and SWEEP are similar) showing the effect of memory

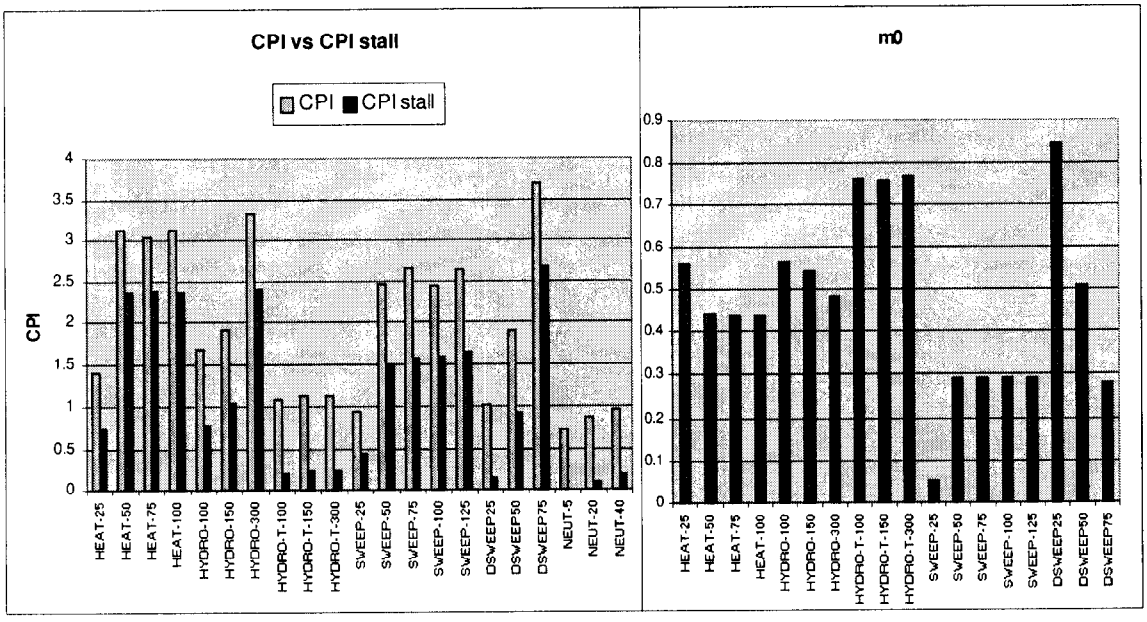

Figure 10. Memory stall & overlap parameters (PowerChallenge)

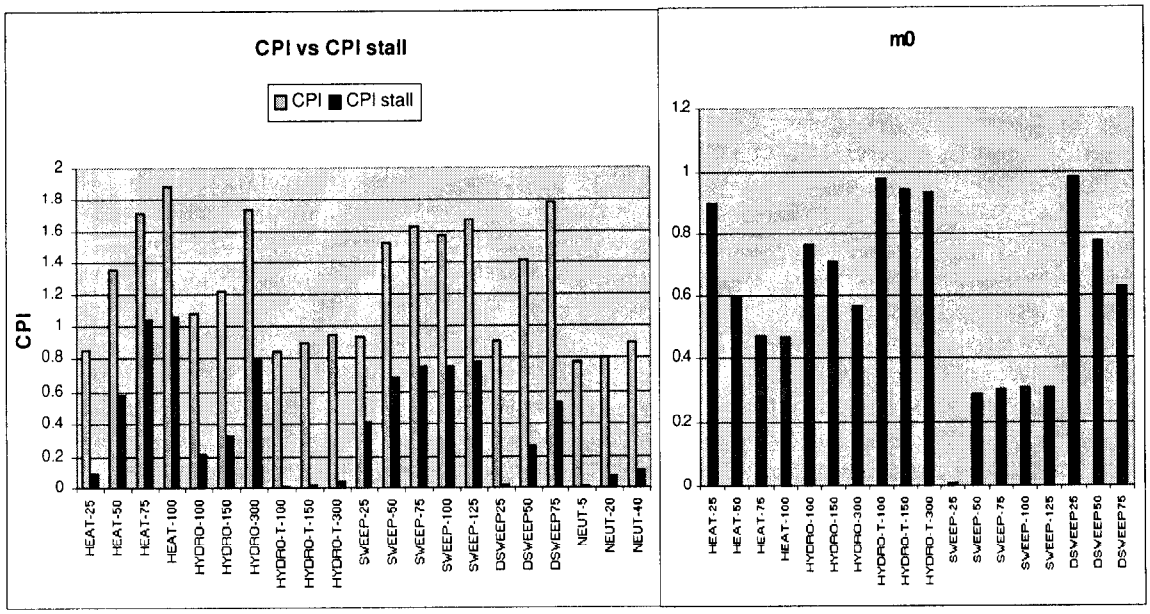

Figure 11. Memory stall & overlap parameters (Origin2000)

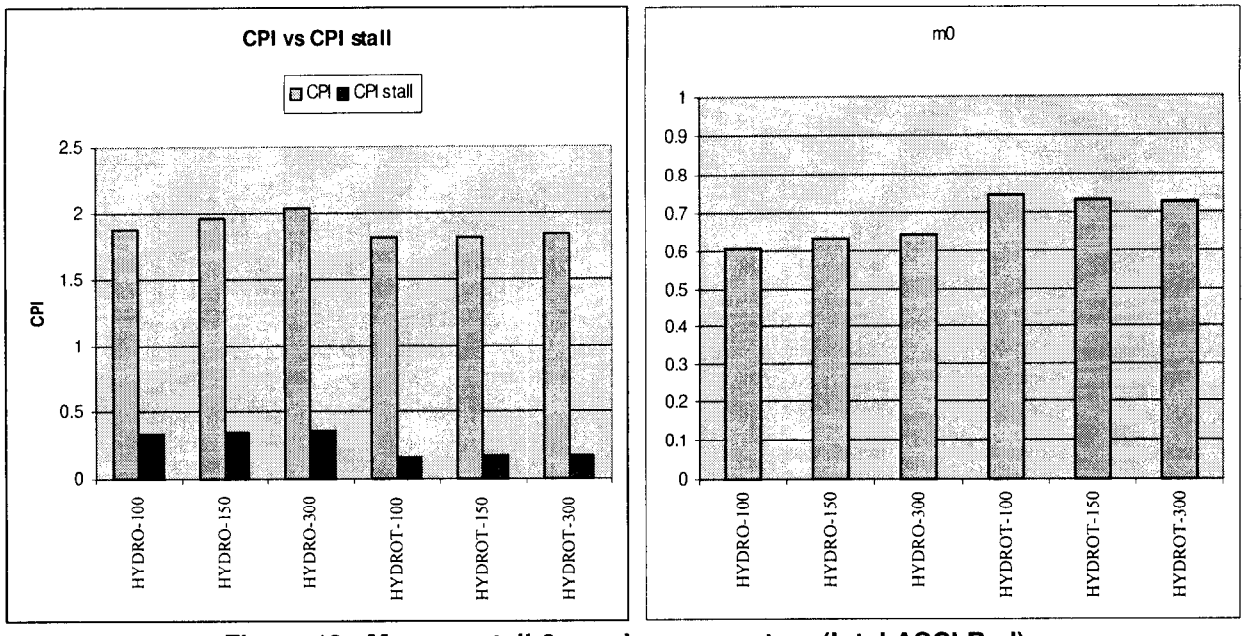

Figure 12. Memory stall & overlap parameters (Intel ASCI Red)

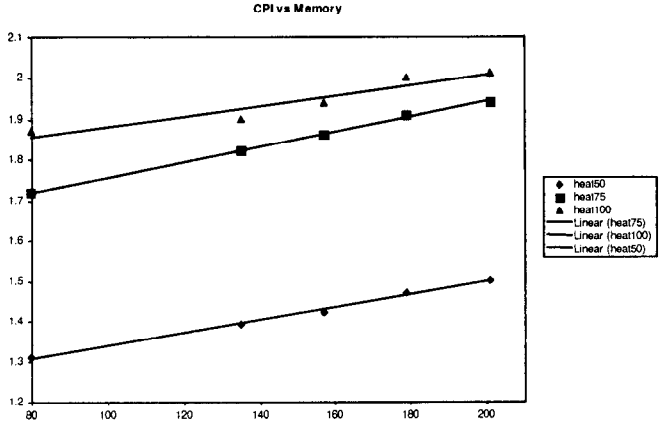

Figure 13. Observed HEAT CPI vs. Memory latency (Origin 2000)

latency on the measured cpi. A linear dependency is observed in agreement with Eq. 4, where the slope is bounded by h_m(see discussion in Section 4).

Using these measurements and other empirical data on the two machines, we can infer the separate the contribution of cache size, memory latency and number of outstanding misses to the improved cpi of the Origin over the PowerChallenge. Let F be a measure of this overall improvement:

$$F = cpi^{PC} / cpi^{O} \qquad (5)$$

We wish to find the contributing factors, f_c, f_o, and f_m (corresponding to cache, outstanding misses and memory latency, respectively) such that:

$$F = f_c * f_o * f_m . \qquad (6)$$

These factors can be defined as follows:

$$f_c = \frac{h_2^{PC} t_2^{PC} + h_m^{PC} t_m^{PC} + cpi_0}{h_2^{O} t_2^{PC} + h_m^{O} t_m^{PC} + cpi_0} \qquad (7)$$

$$f_o = \frac{h_2^O t_2^{PC} + h_m^O t_m^{PC} + cpi_0}{cpi^{O*}} \qquad (8)$$

$$f_m = \frac{cpi^{O*}}{h_2^O t_2^O + h_m^O t_m^O + cpi_0}. \qquad (9)$$

The denominator in f_c can be viewed as the *cpi* of a virtual machine whose characteristics are identical to those of the PowerChallenge but with L2 cache size equal to that of the Origin (4MB L2). The larger cache size simply changes the hit ratios, h_i^{PC}, to h_i^O. Similarly, the denominator in f_o represents the *cpi* of a virtual machine identical to the Origin except for a memory latency equal to that of the PowerChallenge. The cpi^{O*} for this machine is measured as illustrated in Figure 13 (when the memory latency is around 201 cycles). The quantity, f_c, then, is the ratio of the actual PowerChallenge to a Power Challenge with the Origin's cache. The quantity, f_o, is the ratio of this "larger cache" PowerChallenge to an Origin with larger memory latency. Finally, the quantity, f_m, is the ratio of this "large latency" Origin to the real Origin. The separate factors satisfy the relationship in Eq.6. Each of these factors is listed in Table 3, along with the calculated and observed values, F, for the codes. The calculated and observed speedups are in good agreement. With the exception of HYDRO and a small HEAT problem, the values of f_c are 1.0-1.1 indicating that the effect of a larger L2 cache is negligible. The values of f_m are also quite small (most showing 10% or less improvement). The overall improvement for over half of the benchmark codes comes from the increased number of outstanding misses on the Origin. About 75% of the total improvement of the larger HEAT problems and 50% to 80% of SWEEP come from this feature.

Table 3. Observed and calculated performance on the Origin2000

Code	f_c	f_o	f_m	F_{calc}	F_{obs}
HEAT50	1.47	1.40	1.13	2.34	2.37
HEAT75	1.02	1.58	1.09	1.76	1.80
HEAT100	1.00	1.54	1.12	1.73	1.68
HYDRO100	1.41	1.06	1.02	1.52	1.53
HYDRO150	1.34	1.08	1.09	1.59	1.47
HYDRO300	1.28	1.16	1.31	1.94	1.67
HYDRO-T100	1.16	1.05	0.99	1.21	1.28
HYDRO-T150	1.09	1.10	1.03	1.23	1.25
HYDRO-T300	1.01	1.12	1.08	1.22	1.21
SWEEP50	1.05	1.28	1.13	1.52	1.60
SWEEP75	1.00	1.18	1.27	1.50	1.63
SWEEP100	1.00	1.42	1.06	1.52	1.55

7. Conclusions

This paper describes a methodology using a simple memory model with empirical parameters that accounts for the overlap of single processor execution with memory accesses. This method is applied to real applications using performance counter data available on actual machines. In general, this model quantifies the amount of overall time that is spent on memory accesses for each application. On the PowerChallenge, the memory access time can be as large as 80% of the overall execution time. On the Origin 2000, the memory access time is less than 40%. Probably not directly comparable due to inherent architecture differences, the Intel ASCI Red showed stall times less than 20% for the limited codes provided; we would expect these to grow significantly for codes such as SWEEP3D. Using this model, we discover that the increased number of outstanding misses in the Origin 2000 is a major contributing factor to the performance improvement in two out of four codes. The effect of cache size on the performance of these codes is generally much less important except for a code with poor cache reuse (HYDRO). Currently, the methodology is an excellent diagnostic tool that can provide information about the

actual time that an application spends in memory accesses. This model can be eventually incorporated into a performance tool. The methodology can also be applied to any architecture with the necessary hardware counter information. Further work in our group is currently underway to model other microprocessor systems such as the IBM RS/6000 and to increase the spectrum of observed codes for the Intel ASCI Red machine. Future work will attempt to enhance the predictive capability of the model and to develop a software performance tool based on this methodology.

Acknowledgements

We thank Jim Schwarzmeier (SGI/Cray) who reviewed the paper and contributed many thoughtful suggestions for improvement and future work. Also we extend thanks to Pat Fay of Intel for insight into the workings of the ASCI Red performance counters.

References

[1] Wulf, W. A. and McKee, S. A. "Hitting the Memory Wall: Implications of the Obvious," Computer Architecture News, Association for Computing Machinery, March, 1995.

[2] Burger, D. C., Goodman, J. R., and Kagi, A., "The Declining Effectiveness of Dynamic Caching for General-Purpose Microprocessors," Univ. Wisconsin Computer Sciences Tech. Report CS-TR-95-1261, Jan. 1995, and references therein.

[3] Galles, M. and Williams, E., "Performance Optimizations, Implementation, and Verification of the SGI Challenge Multiprocessor," Silicon Graphics Computer Systems, ," Silicon Graphics Computer Systems, Mountain View, CA web paper http://www.sgi.com/Technology/challenge_paper.html.

[4] Laudon, J. and Lenowski, D., "The SGI Origin: A ccNUMA Highly Scalable Server," Proc. Compcon Spring.1997, IEEE Computer Society, Los Alamitos, California.

[5] (a) MIPS Technologies, Inc., "R10000 Microprocessor Product Overview," MIPS Product Preview, 1995. (b) Yeager, K. C., "The MIPS R10000 Superscalar Microprocessor," IEEE Micro, April, 1996, pp 28-40.

[6] Zagha, M., Larson, B., Turner, S., and Itzkowitz, M., "Performance Analysis Using the MIPS R10000 Performance Counters," Proc. Supercomputing '96, IEEE Computer Society, Los Alamitos, California, 1996.

[7] Luo, Y., Lubeck, O.M., and Wasserman, H. J., "Preliminary Performance Study of the SGI Origin2000," Los Alamos National Laboratory Unclassified Release LA-UR -, 1997.

[8] Koch, K. R., Baker, R. S. and Alcouffe, R. E., "Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor," Trans. of the Amer. Nuc. Soc., 65, 198, 1992.

[9] W. D. Schulz, "Two-Dimensional Lagrangian Hydrodynamic Difference Equations," Methods in Computational Phys. Vol 3, p1, 1964.

[10] McVoy, L. and Staelin, C., "lmbench: Portable Tools for Performance Analysis," http://reality.sgi.com/lm_engr/lmbench.

[11] Vernon, M.V, Lazowska, E. D., and Zahorjan, J., "An Accurate and Efficient Performance Analysis Technique for Multiprocessor Snooping Cache-consistency Protocols," in Proc. 15th Annu. Symp. Comput. Architecture, Honolulu, HI, June, 1988, pp 308-315.

[12] Larson, B., Silicon Graphics Computer Systems, private communication, January, 1997.

[13] Turner, S., Silicon Graphics Computer Systems, private communication, January, 1997.

[14] Bhandarkar, D. and Cvetanovic, Z., "Performance Characterization of the Alpha 21164 Microprocessor Using TP and SPEC Workloads," Proc. Second. Int. Sypm. on High-Perf. Comp. Arch., IEEE Computer Society Press, Los Alamitos Ca., 1996.

[15] Bhandarkar, D. and Ding, J., "Performance Characterization of the Pentium Pro Processor, " Proc. Third. Int. Sypm. on High-Perf. Comp. Arch., IEEE Computer Society Press, Los Alamitos Ca., pp 288-297, 1997.

[16] Mattson, T. and Henry, G., "An Overview of the Intel TLFOPS Supercomputer", Intel Technical Journal, Jan. 1998.

[17] Sandia National Lab, "The ASCI Red TFLOPS Supercomputer", http://www.sandia.gov/ASCI/Red/, Nov. 1996.